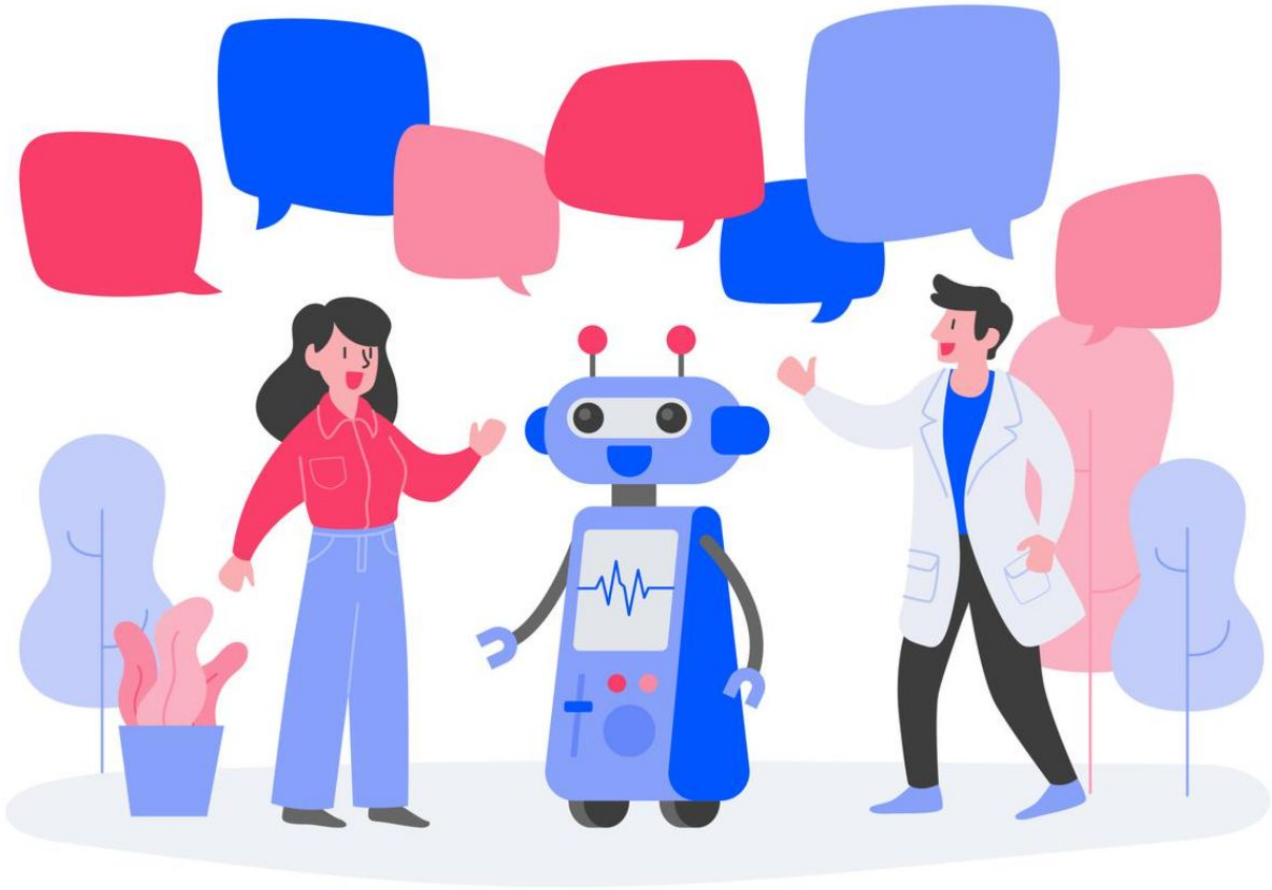
ETH zürich

Natural Language Processing
Project 2

Alizée Pace
alizee.pace@ai.ethz.ch
29.03.2021

Motivation: Give you a taste for the usefulness of NLP in medicine!

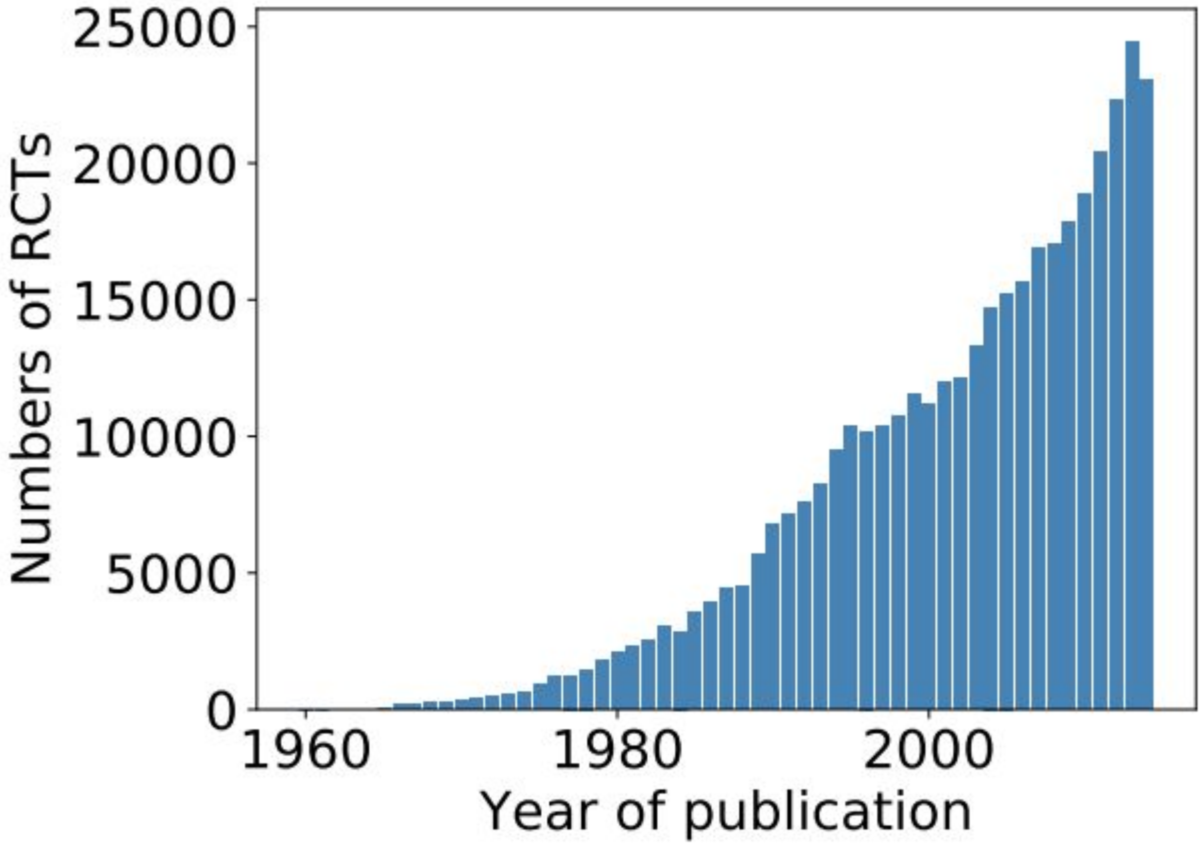


Language: succinct, efficient means of transmitting information.

- Endless sources of information in medical text:
  - 🏥 patient records & letters
  - 👩‍⚕️ doctor notes
  - 💬 conversation transcripts
  - 💰 insurance claims
- Privacy and ethics issues with most of these. Let's focus on publicly available, non-identifiable data:
  - 📚 **medical literature**
  - 📹 journalism
  - 🗣️ social media

Source: Silvia Natalia on vecteezy.

The medical literature is huge…



- Randomized Controlled Trials (RCTs): best source of medical evidence.
- >1M published so far: **Challenging to efficiently parse the existing literature!**

Source: Dernoncourt & Lee, IJCNLP 2017.

Some attempts to facilitate literature parsing

- literature reviews
- graphical abstracts
- clear outlining of results
- **structured abstracts** (~50% of literature)

Dataset: PubMed 200k RCT

[Franck-Dernoncourt/pubmed-rct]

- ~200,000 PubMed abstracts of randomized controlled trials, totaling 2.3 million sentences.
- Designed for sequential sentence classification ➡️ Useful to facilitate literature reviewing.

  Labels: background, objective, method, result, conclusion.

- Other interesting use cases:

  📚 automatic text summarization

  💡 information extraction, e.g. what is the scientific claim in this abstract?

  🧐 information retrieval, e.g. what is the effect of X drug on Y cancer patients?

- 3 data splits: **Train** (model training); **Dev** (hyperparameter selection); **Test** (final model testing).

Data Format & Preprocessing Example

normalisation?

'BACKGROUND\tThe emergence of HIV as a chronic condition means that people living with HIV are required to take more responsibility for the self-management of their condition , including making physical , emotional and social adjustments .'

label: BACKGROUND

sentence: 'The emergence of HIV…'

`.lower()` `.split(' ')`

lowercasing & tokenisation

[~~'the'~~, 'emergence', ~~'of'~~, 'hiv', ~~'as'~~, 'a', 'chronic', 'condition', 'means', ~~'that'~~, 'people', 'living', ~~'with'~~, 'hiv', ~~'are'~~, 'required', ~~'to'~~, 'take', ~~'more'~~, 'responsibility', ~~'for'~~, ~~'the'~~, 'self-management', ~~'of'~~, 'their', 'condition', ~~','~~, 'including', 'making', 'physical', ~~','~~, 'emotional', ~~'and'~~, 'social', 'adjustments', ~~'.'~~]

`nltk.corpus.stopwords.words('english')` `string.punctuation`

stop-words & punctuation removal

What about numbers?

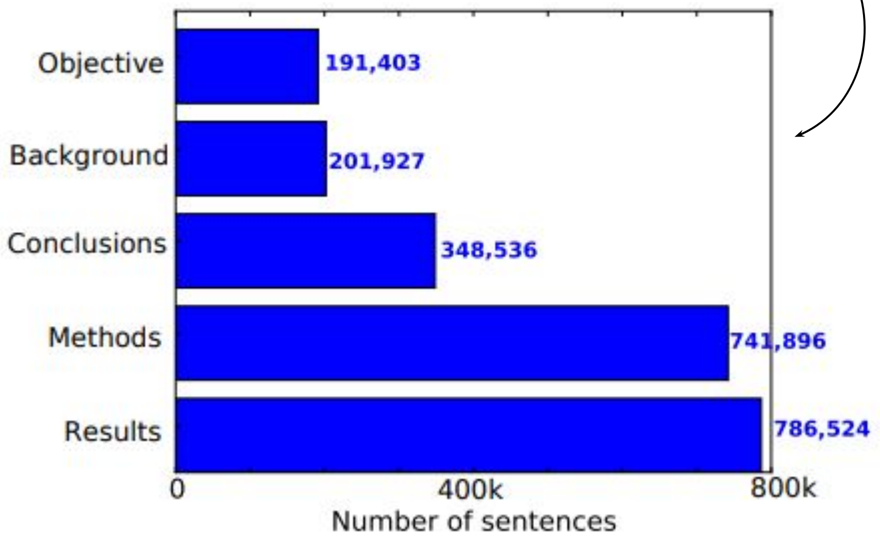
['emergence', 'hiv', 'a', 'chronic', 'condition', 'means', 'people', 'living', 'hiv', 'required', 'take', 'responsibility', 'the', 'self-management', 'their', 'condition', 'including', 'making', 'physical', 'emotional', 'social', 'adjustments']

optional: stemming, lemmatisation?

+ why do/don't they help?

Sequential Sentence Classification

some class imbalance



- Multiclass classification
- **Evaluation metric**: F1-score (`weighted`).
- Analyse the **confusion matrix** between output classes obtained with your best-performing model.

TASK 1: BASELINE MODEL

documents: sentences or abstracts in this task?

- **Preprocessing**: lowercasing, stop-words & punctuation removal etc.
- Obtain sentence embeddings through **TF-IDF** (e.g. sklearn `TfidfVectorizer`).
- Train classifier to predict corresponding abstract class.

Source: Dernoncourt & Lee, IJCNLP 2017.

Word Embeddings

How can we translate the *lexical + semantic meaning of words* to a *numerical entity*?

🤴 - 👨 + 👩 = 👸

king - man + woman = queen

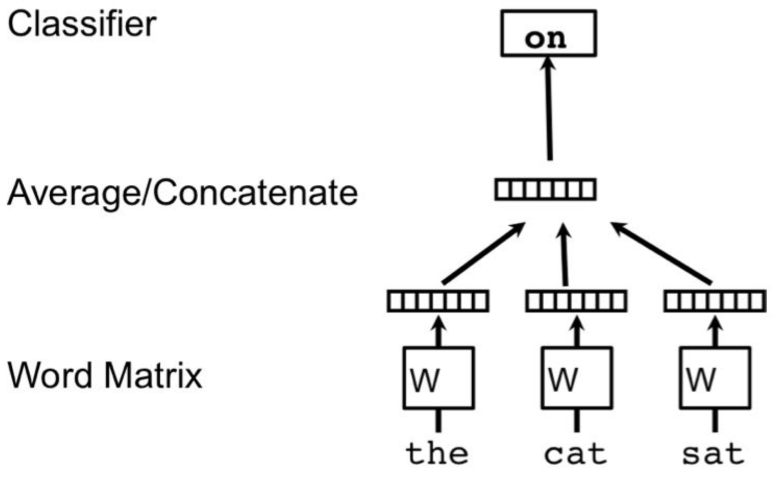
Word2Vec: word embeddings depends on **context**.

CBOW: predict word from context. *Skip N-gram*: predict context from word.

FastText: similar but with sub-word decompositions. Allows to embed unseen words.

TASK 2: WORD EMBEDDING

- Train a word embedding model such as **Word2Vec** or **FastText** (we recommend using the `gensim` library).
- Obtain sentence embeddings by averaging or concatenating word embeddings.
- Train classifier to predict corresponding abstract class.



Optional: discuss any interesting semantic relationships between word embeddings.

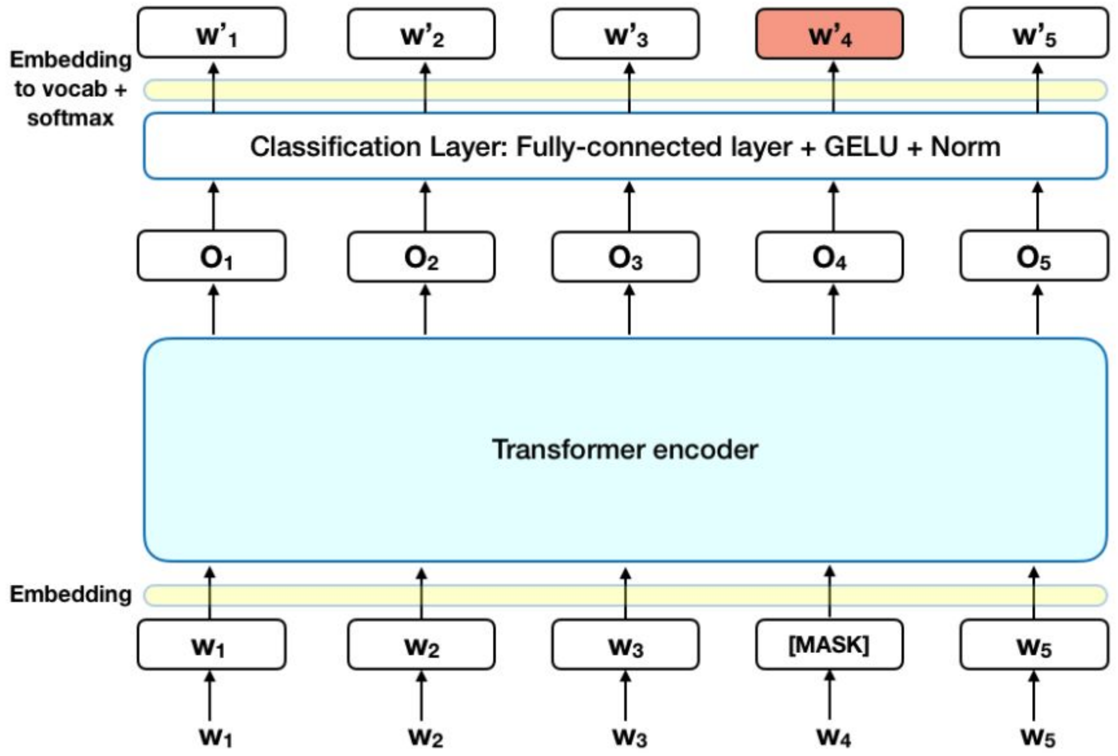
Transformer-Based Language Models

better performance than left-to-right LM

Recap on **BERT** (Bidirectional Encoder Representations from Transformers, Devlin et al., 2018):

- Encoder of Transformer architecture.
- Trained via Masked LM (right) and Next Sentence Prediction.

BERT pre-trained on biomedical language can be found on HuggingFace. Think about what data the model should be pre-trained on (MIMIC, PubMed, etc.).



TASK 3: TRANSFORMER MODEL

Evaluate the performance of **BERT** on the given task. See mini-tutorial in additional slides.

- Pre-trained, no fine-tuning.
- Pre-trained and fine-tuned on our dataset. Which parameters/layers must be tuned?

Deliverables

- Solve all tasks.
- **Report** of max. 4 pages, 11pt (+ 1 page for references + 1 page of appendix if needed).
- **Well-commented code/jupyter notebooks** with conda environment and README.
- Do not hardcode any results! We will run your code.
- Ensure sequential execution and **reproducibility**.
- **Do not copy solutions from previous projects!** We are aware of all existing solutions on github. We run code similarity checks and check for plagiarism in the reports from previous years solutions. Any plagiarism will result in a 0 grade for all projects.
- **Deadline**: 25.04.2022

Grade

- To grade the project we will focus (on equal parts) on:
  - the content, organisation, clarity, quality and writing of the **final report**.
  - the quality of the **implementation** (reproducibility and clarity).
  - the **creativity/performance*** of the methods used to solve the tasks, and the reasons behind the choices.
- The **prerequisites** to get the maximum grade are:
  - write a clear and **good report.**
  - submit a **clean code** with **easy instructions** on how to reproduce each result of the report.
  - solve every task with **well-justified** methods.
  - bonus: implement **creative** models for one task (e.g. alternative embedding or language model)

*We will consider resource constraints. Aside from correct baseline implementation, the aim is not to get the best performance but to explore and discuss relevant methods.

Questions?

Also on Moodle (preferred: your classmates probably have similar questions!)
or by email at alizee.pace@ai.ethz.ch.

Additional Slides

TF-IDF

In **TF**, all terms are equally important. **TF-IDF** scales down the weight of frequently-occurring terms.

via Document Frequency (**DF**) = (# of documents that contain a term *t*) / (total # of documents)

- high TF-IDF weight when term occurs *many times* within *few* documents ➡️ highly relevant!
- lower TF-IDF weight when term occurs *rarely* or in *many* documents.

Useful to extract essential **keywords** from text documents.

Pre-trained BERT model

Full tutorial on [HuggingFace](#) 🤗
(+ other *How-to guides*)

which AutoModel should you use? i.e. what task are we solving?

replace with pretrained weights of interest

```
from transformers import AutoTokenizer, AutoModel

tokenizer = AutoTokenizer.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
model = AutoModel.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
```

NB. Pytorch and Tensorflow-compatible interfaces are available if preferred.